STATISTICAL ARBITRAGE · บทที่ 9

Regime-Switching OU: ตลาดมีหลาย 'สภาวะ'

Markov chain 3 สภาวะ — Normal / Stressed / Broken — ปรับ OU parameters ตาม regime

ตัวอย่างหลัก: Bybit BTC Perp ↔ Lighter BTC Perp

แก่นของบทนี้ — อ่านก่อน

ตลาดไม่ได้อยู่ใน "สภาพเดียว" ตลอดเวลา — ช่วง Normal spread กลับเร็ว ช่วง Stressed volatility พุ่ง ช่วง Broken spread ไม่กลับเลย ต้องรู้ว่าอยู่ใน regime ไหน แล้ว **เทรดเฉพาะ Normal regime**

κ, θ, σ เปลี่ยนตาม regime ที่ตลาดอยู่

9.1 ปัญหาของ Single-OU

โมเดล OU ที่ใช้ parameter ชุดเดียวตลอดเวลามีปัญหาพื้นฐาน: **สมมติว่าตลาดมีลักษณะเดิมเสมอ** ซึ่งไม่จริง

สามอาการที่บอกว่า Single-OU ไม่เพียงพอ

- **ช่วง volatile:** σ พุ่งขึ้น 3–5x จากค่าปกติ — signal เดิมมี noise มากเกินไป เข้า position แล้วขาดทุน
- **ช่วง broken:** spread ไม่กลับมาหา θ เลยแม้ผ่านไปหลายชั่วโมง — κ ลดลงจนใกล้ศูนย์
- **ช่วง calm เกินไป:** spread แทบไม่ขยับ — opportunity หมด แต่ model ยังส่ง signal

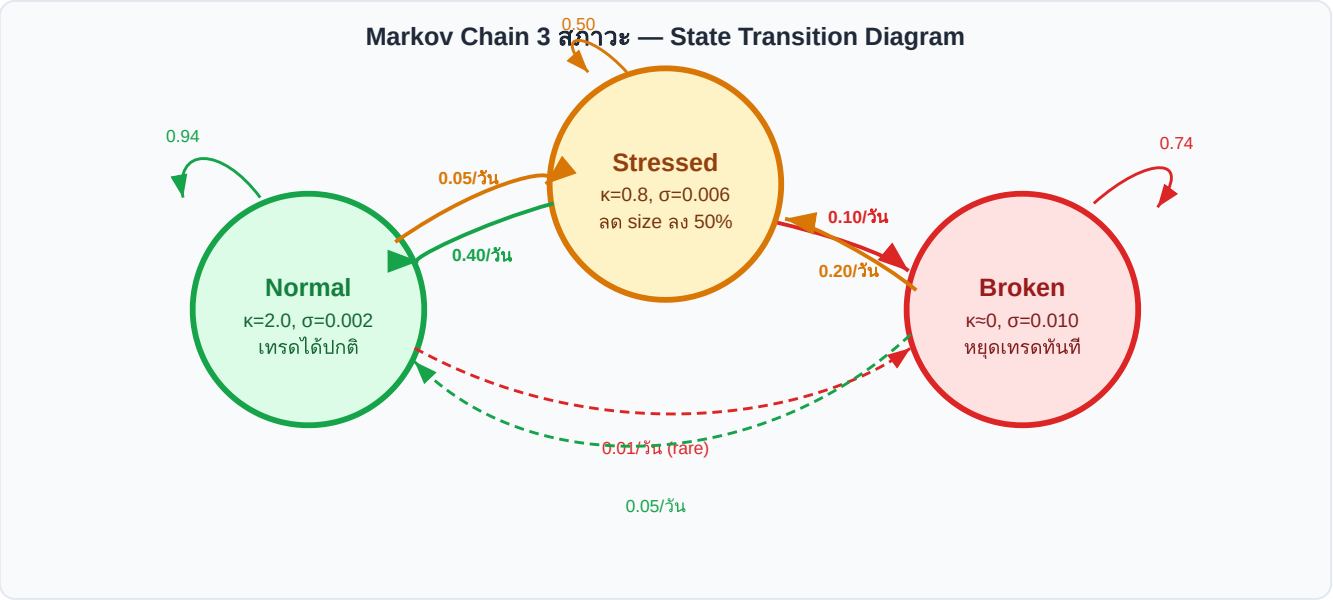
ผลที่ตามมาเมื่อใช้ Single-OU ในทุก regime

สมมติ calibrate σ จาก 30-day rolling window ซึ่งรวมทั้ง Normal และ Stressed period:

- σ_estimated จะสูงกว่าความเป็นจริงใน Normal regime → entry threshold สูงเกิน → พลาด opportunity
- σ_estimated จะต่ำกว่าความเป็นจริงใน Stressed regime → stop-loss แคบเกิน → ถูก stop out บ่อย
- ผลลัพธ์: กลยุทธ์ทำงานได้ไม่ดีทั้งสอง regime

9.2 3-State Markov Chain

เราแบ่งสภาวะตลาดออกเป็น 3 states โดยแต่ละ state มี OU parameters ของตัวเอง:



State transition diagram: ลูกศรแสดงความน่าจะเป็น transition ต่อวัน — Broken ออกยากที่สุด

Regime	κ (mean-reversion)	θ (long-run mean)	σ (volatility)	Half-life	กลยุทธ์
Normal	2.0 (เร็ว)	≈ 0 (spread กลับศูนย์)	0.002 (ต่ำ)	~8 ชั่วโมง	เทรดเต็ม size
Stressed	0.8 (ช้า)	≈ 0 (ยังกลับ แต่ช้า)	0.006 (สูง)	~21 ชั่วโมง	ลด size 50%
Broken	≈0 (ไม่ revert)	undefined (drift)	0.010 (สูงมาก)	∞ (ไม่กลับ)	หยุดเทรดทันที

θ ≈ 0 ใน Normal/Stressed หมายความว่า spread ยังกลับสู่ศูนย์ (หรือ baseline); ใน Broken regime κ → 0 ทำให้ θ ไม่มีความหมาย — spread drift ออกไปโดยไม่มีแรงดึงกลับ

9.3 Hamilton Filter

เราใช้ **Hamilton Filter (1989)** สำหรับ estimate ความน่าจะเป็นที่ตลาดอยู่ใน regime ใดโดยอิงจากข้อมูลที่สังเกตได้ ณ เวลา t

สัญชาตญาณ Hamilton Filter — 2 ขั้นต่อ bar

1. **Predict:** "ถ้าเมื่อวานอยู่ใน regime j ด้วยความน่าจะเป็น p_j วันนี้น่าจะอยู่ใน regime ใด?" — ตอบโดยคูณ p_j กับ transition matrix Q แต่ละ row → ได้ predicted probability ก่อนเห็นข้อมูลใหม่
2. **Update:** "spread วันนี้มีค่า y_t — regime ไหนที่ explain ค่านี้ได้ดีที่สุด?" — ตอบโดยคูณ predicted probability × likelihood ที่ y_t จะเกิดขึ้นในแต่ละ regime แล้ว normalize ให้บวกเป็น 1

ทำซ้ำทุก bar → ได้ P(regime | ข้อมูลถึงวันนี้) แบบ real-time โดยไม่ต้องรู้ regime จริงล่วงหน้า

HAMILTON FILTER — สมการหลัก

`P(S_t=j | Y_t) ∝ f(y_t | S_t=j) · Σ_i P(S_t=j | S_{t-1}=i) · P(S_{t-1}=i | Y_{t-1})`

S_t = hidden state (Normal=0, Stressed=1, Broken=2) ณ เวลา t
Y_t = ข้อมูลที่สังเกตได้ถึงเวลา t (residuals ทั้งหมด)
f(y_t | S_t=j) = likelihood ของ observation ถ้าอยู่ใน regime j
P(S_t=j | S_{t-1}=i) = transition probability (จาก matrix Q)
∝ = proportional to (normalize ให้ผลรวมเป็น 1)

Likelihood Function สำหรับแต่ละ Regime

ถ้า regime j มี OU parameters (κ_j, θ_j, σ_j) likelihood ของ observation y_t คือ:

```
f(y_t | S_t=j, y_{t-1}) = Normal(y_t; mu_j, var_j) mu_j = y_{t-1} * exp(-kappa_j * dt) + theta_j * (1 - exp(-kappa_j * dt)) var_j = (sigma_j^2 / (2*kappa_j)) * (1 - exp(-2*kappa_j * dt))
```

นี่คือ exact OU transition density: ถ้า y_{t-1} รู้แล้ว y_t ต่อไป Normal distribution ที่ mean และ variance คำนวณได้จาก OU parameters

Transition Matrix Q — ตัวอย่าง

```
Normal Stressed Broken Normal [ 0.94 0.05 0.01 ] Stressed[ 0.40 0.50 0.10 ]
Broken [ 0.05 0.20 0.75 ] แต่ละ row บวกกันได้ 1 (must sum to 1) Q[i,j] = P(S_t=j
| S_{t-1}=i)
```

9.4 Pseudo-code: regime_filter()

```
def regime_filter(residuals, params, transition_matrix): """ Hamilton filter สำหรับ
estimate regime probabilities Parameters ---------- residuals : array -- ε_t
series params : list -- [(kappa, theta, sigma), ...] per regime transition_matrix
: 2D array -- Q matrix, shape (n_regimes, n_regimes) Returns ------- probs : 2D
array -- shape (T, n_regimes) probs[t, j] = P(S_t=j | Y_t) """ n_regimes =
len(params) T = len(residuals) # เริ่มด้วย uniform prior probs =
initialize_probs(n_regimes) # [1/3, 1/3, 1/3] all_probs = zeros((T, n_regimes))
all_probs[0] = probs for t in range(1, T): # Step 1: log-likelihood ของ y_t ในแต่ละ
regime # ✅ FIX: ทำงานใน log-space เพื่อป้องกัน numerical underflow # เมื่อ T ยาวหรือ
likelihood เล็กมาก การคูณซ้ำๆ ทำให้ค่าเป็น 0.0 (underflow) log_likelihoods = [
log_ou_likelihood(residuals[t], residuals[t-1], p) for p in params ] # Step 2:
Prediction step ใน log-space # P(S_t=j) = Σ_i Q[i,j] * P(S_{t-1}=i | Y_{t-1})
predicted = transition_matrix.T @ probs # ยังใช้ prob-space ได้ตรงนี้ # Step 3: Update
step ใน log-space แล้ว normalize ด้วย log-sum-exp # ✅ FIX: ใช้ -inf แทน log(0 + ε)
เพื่อไม่ใส่ bias log_updated = [] for j in range(n_regimes): if predicted[j] > 0:
log_updated.append(log_likelihoods[j] + log(predicted[j])) else:
log_updated.append(-inf) # regime นี้เป็นไปไม่ได้ # log-sum-exp trick: log(Σ e^{x_i}) =
max_x + log(Σ e^{x_i − max_x}) # เหตุผล: x_i − max_x ≤ 0 เสมอ ดังนั้น e^{x_i−max_x} ∈
(0,1] → ไม่ overflow max_log = max(log_updated) if max_log == -inf: # ทุก regime
impossible → fallback uniform probs = [1.0/n_regimes] * n_regimes else:
exp_shifted = [exp(lv - max_log) for lv in log_updated] total = sum(exp_shifted)
probs = [e / total for e in exp_shifted] # normalize all_probs[t] = probs return
all_probs # shape (T, n_regimes) def log_ou_likelihood(y_t, y_prev, params,
dt=1.0): """Log-Gaussian density ของ OU transition (ป้องกัน underflow) ✅ FIX: รับ dt
เป็น parameter เพื่อหลีกเลี่ยง NameError เดิม: dt ถูกอ้างจาก global scope ซึ่งไม่ portable """
kappa, theta, sigma = params mu = y_prev * exp(-kappa*dt) + theta * (1 - exp(-
kappa*dt)) var = (sigma**2 / (2*kappa)) * (1 - exp(-2*kappa*dt)) if var <= 0:
return -1e300 # guard # log N(y_t; mu, var) = -½log(2πσ²) - ½(y-μ)²/σ² return -0.5
* log(2 * pi * var) - 0.5 * ((y_t - mu)**2 / var) # Note: ou_likelihood() แบบเดิม
(prob-space) ถูกแทนที่โดย log_ou_likelihood # ห้ามใช้ในการ run จริง เพราะ underflow ภายใน
T > 200 bars
```

ข้อสังเกตสำคัญ

- Filter ทำงาน online (real-time) — ทุก tick ใหม่อัปเดต probs ได้ทันที
- ไม่ต้องรู้ true state — แค่ observe residuals แล้ว infer
- Transition matrix Q ต้อง calibrate จาก historical data (EM algorithm)
- เมื่อ probs เปลี่ยนแปลงเร็วมาก → อาจเป็น signal ของ jump (บทที่ 8)

9.5 Trade Rule: เทรดเฉพาะ Normal Regime

กฎหลักของกลยุทธ์ regime-switching:

TRADE RULE

เปิด position ใหม่ได้เมื่อ **P(Normal) > 0.70** เท่านั้น

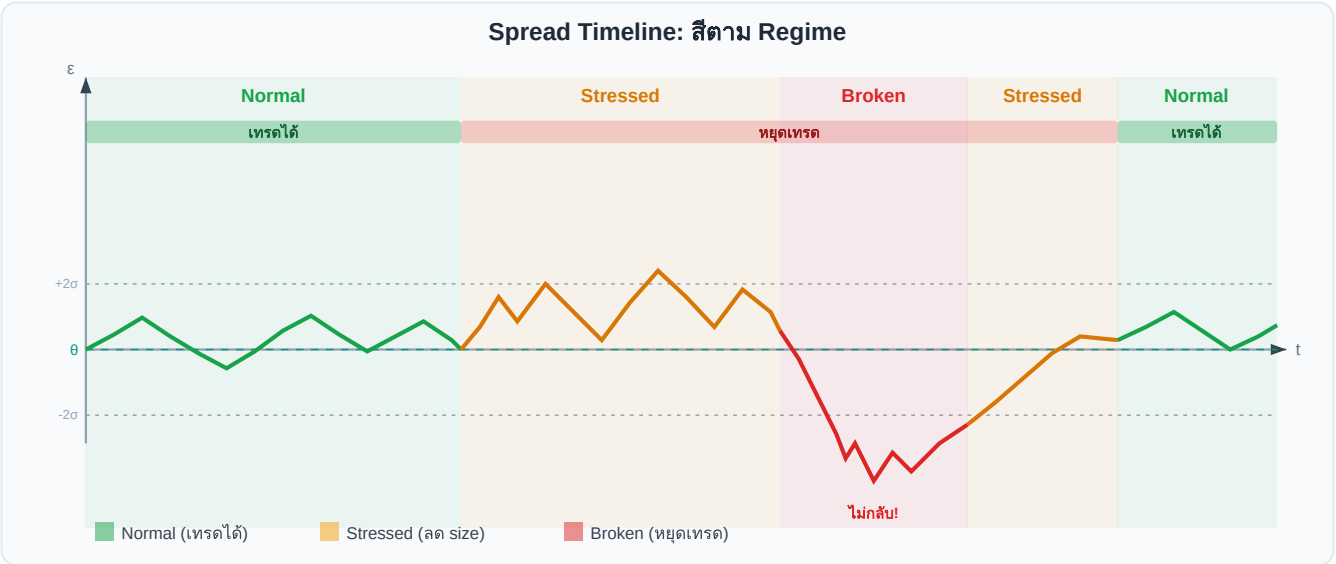
`Trade if P(S_t = Normal | Y_t) > 0.70`

Regime-dependent sizing

```
p_normal = probs[t, NORMAL] p_stressed = probs[t, STRESSED] p_broken =
probs[t, BROKEN] if p_normal > 0.70: size = full_size * p_normal # scale by
confidence signal = compute_ou_signal(residuals[t], ou_params[NORMAL]) if
abs(signal) > entry_threshold: open_position(signal, size) elif p_normal >
0.50: # Borderline — ลด size มาก size = full_size * 0.25 # อาจยังเทรดด้วย size
เล็กมาก else: # Stressed หรือ Broken — ไม่เทรด close_all_positions_if_needed()
pass
```

P(Normal)	P(Stressed)	P(Broken)	Action
> 0.70	< 0.25	< 0.10	เทรดเต็ม size
0.50–0.70	any	< 0.10	เทรด 25% size
< 0.50	> 0.40	any	หยุดเทรด (Stressed)
any	any	> 0.20	ปิด position ทั้งหมด (Broken)

9.6 Timeline Diagram: Spread Colored by Regime



สีบนเส้น spread: เขียว=Normal (เทรดได้), เหลือง=Stressed (ระวัง), แดง=Broken (หยุดทันที)

9.7 Running Example: Bybit↔Lighter — Regime ในทางปฏิบัติ

Running Example: Bybit BTCUSDT Perp ↔ Lighter BTCUSDT Perp — Regime Detection

Normal Regime — ช่วง Low Funding Variance

เมื่อ funding rate ทั้งสองตลาดมีความผันผวนต่ำ (ต่างกัน <0.005% ต่อชั่วโมง):

- P(Normal) ≈ 0.85 → เทรดได้เต็ม size
- Half-life ≈ 6–10 ชั่วโมง → position เปิดปิดรวดเร็ว
- σ ของ spread ≈ 0.002 → entry ที่ ±2σ ≈ ±$130 บน $65,000 BTC

Stressed Regime — ช่วงก่อน Listing Events

ช่วง 24–48 ชั่วโมงก่อน token listing ใหม่บน Bybit:

- Funding rate เริ่ม volatile → spread ขยับมากขึ้น
- P(Normal) ลดลงจาก 0.85 → 0.45 ภายใน 2 ชั่วโมง
- Filter detect: P(Stressed) = 0.48 → ลด size ลง 50%
- Half-life ยืดออกเป็น 20+ ชั่วโมง → ต้องถือ position นานขึ้น

```
# Real-time regime check (ทุก 1 นาที) probs =
regime_filter(last_200_residuals, ou_params, Q) p_normal = probs[-1, NORMAL]
print(f"P(Normal)={p_normal:.2f}, " f"P(Stressed)={probs[-1,STRESSED]:.2f},
" f"P(Broken)={probs[-1,BROKEN]:.2f}") # Output ช่วง listing event: #
P(Normal)=0.42, P(Stressed)=0.51, P(Broken)=0.07 # → ลด size 75% และตั้ง alert
```

Broken Regime — หลัง Unexpected Delisting

เมื่อ Lighter delists BTC perp โดยไม่แจ้งล่วงหน้า spread ไม่ converge อีกต่อไป — ควร detect ภายใน 15–30 นาที จาก P(Broken) พุ่งขึ้นเกิน 0.3

โจทย์ท้ายบท

โจทย์ 9.1 — ควรเทรดหรือไม่?

ณ เวลา t regime filter ให้ผล:

- P(Normal) = 0.65
- P(Stressed) = 0.30
- P(Broken) = 0.05

Threshold ปัจจุบัน: เทรดเมื่อ P(Normal) > 0.70

ถาม: ควรเทรดหรือไม่? ถ้าไม่ ให้อธิบายว่า ควรทำอะไรแทน? ถ้าใช่ ควรใช้ size เท่าไร?

▶ ดูคำตอบ

โจทย์ 9.2 — Transition Matrix: Row Sum

Transition matrix Q ของระบบ 3 regimes:

```
Q = [[0.94, 0.05, 0.01], [0.40, 0.50, 0.10], [0.05, 0.20, 0.75]]
```

ถาม: Row sum ของ Q ต้องเท่ากับอะไร? เพราะอะไร? ถ้า Row 2 (Stressed) มีค่า [0.40, 0.55, 0.10] จะเกิดอะไรขึ้นและแก้ไขอย่างไร?

▶ ดูคำตอบ

โจทย์ 9.3 — Bybit เปลี่ยน Funding Formula

Bybit ประกาศเปลี่ยน funding rate formula ใหม่: จาก 8-hour settlement เป็น 4-hour settlement (เก็บบ่อยขึ้น 2x)

ถาม: การเปลี่ยนแปลงนี้ส่งผลต่อ regime distribution อย่างไร? และต้องทำอะไรกับ transition matrix Q และ OU parameters?

▶ ดูคำตอบ

อ้างอิงสำคัญ (References)

- **Hamilton, J. D. (1989).** A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica*, 57(2), 357–384. — บทความต้นฉบับ Markov-switching model และ Hamilton filter ที่ Ch.9 ใช้โดยตรง
- **Hamilton, J. D. (1990).** Analysis of time series subject to changes in regime. *Journal of Econometrics*, 45(1–2), 39–70. — ขยาย Hamilton (1989) สู่ continuous-state และ multivariate regime models
- **Kim, C.-J., & Nelson, C. R. (1999).** *State-Space Models with Regime Switching.* MIT Press. — ตำราฐาน Markov-switching กับ OU/state-space; ครอบคลุม Kim smoother ที่ปรับปรุง Hamilton filter

Ch.9 Regime-Switching OU | ต่อไป → **Ch.10: Kalman Filter สำหรับ Dynamic β**